

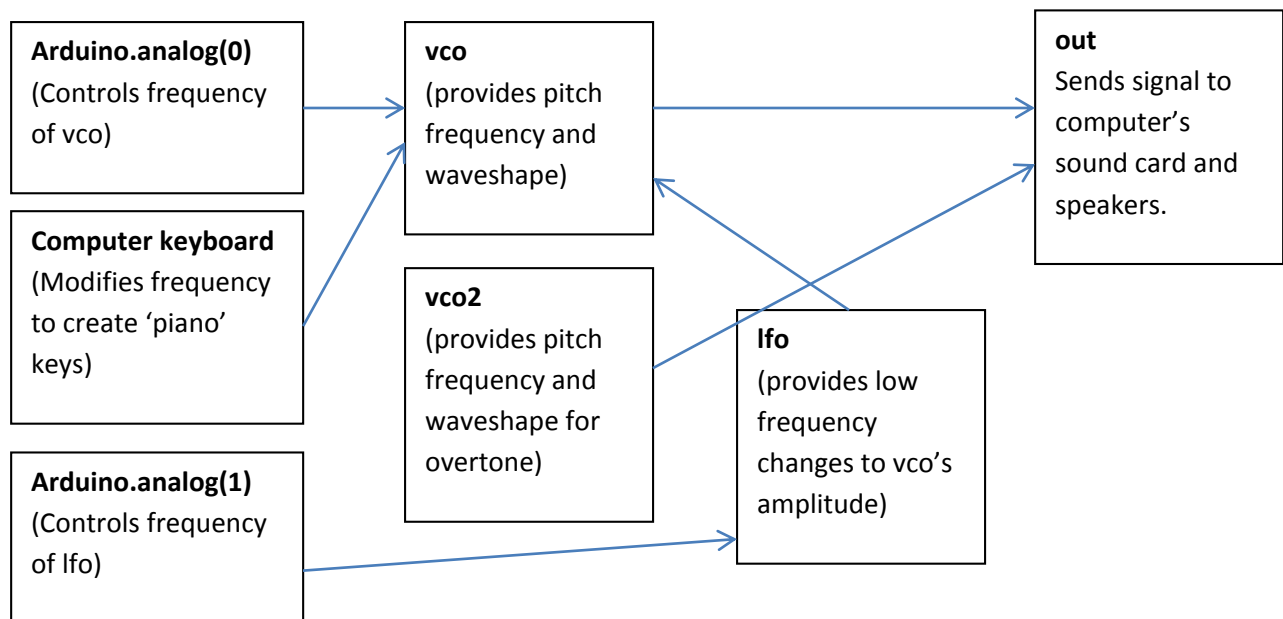
## Arduino and Processing Synthesizer Simulator

### Building and Software Instructions

#### Description:

This lesson will describe the wiring of an Arduino based controller with two potentiometers. These two potentiometers will allow the user to control the frequency of two oscillators. One Oscillator will act as a 'Voltage Control Unit' (VCO) and will provide the pitch for the synthesizer. The second oscillator will act as a 'Low Frequency Oscillator' (LFO). The LFO will patch into the VCO to change the amplitude of the pitch and add effects.

The Processing sketch 'arduinomynth.pde' will simulate an analog synthesizer with virtual oscillator units. A diagram of the setup is shown below:



#### This lesson will contain three sections:

1. Wiring the Arduino Controller
2. Downloading the Arduino Software and Setting up the Arduino Board
3. Copying the Arduino library to the Documents/Processing/libraries folder
4. Download the arduinosynth code and running the code on Processing to simulate the synthesizer
5. Modifying the code to change the sound waveforms, add unit generators, and change patching.

## Process: Section 1

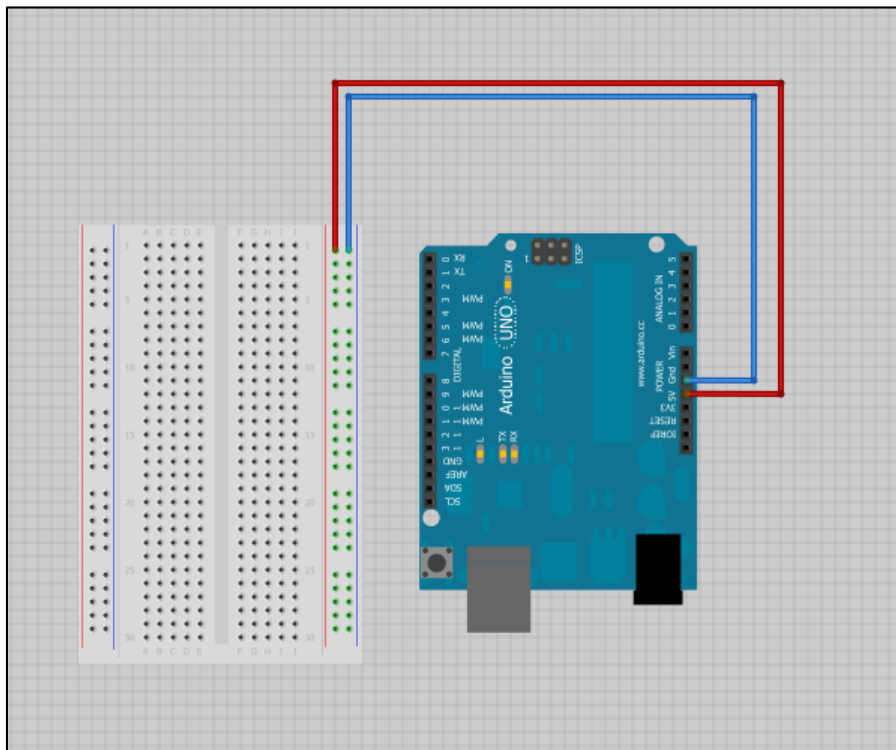
1. You will need the following from the Arduino Kit:
  - a. Arduino Board and Breadboard
  - b. Three Red wires (or orange) for current
  - c. Three Blue wires (or black, green) for ground
  - d. Two white wires (or yellow) for signal
  - e. Two potentiometers (The small blue or white rotating control knobs)



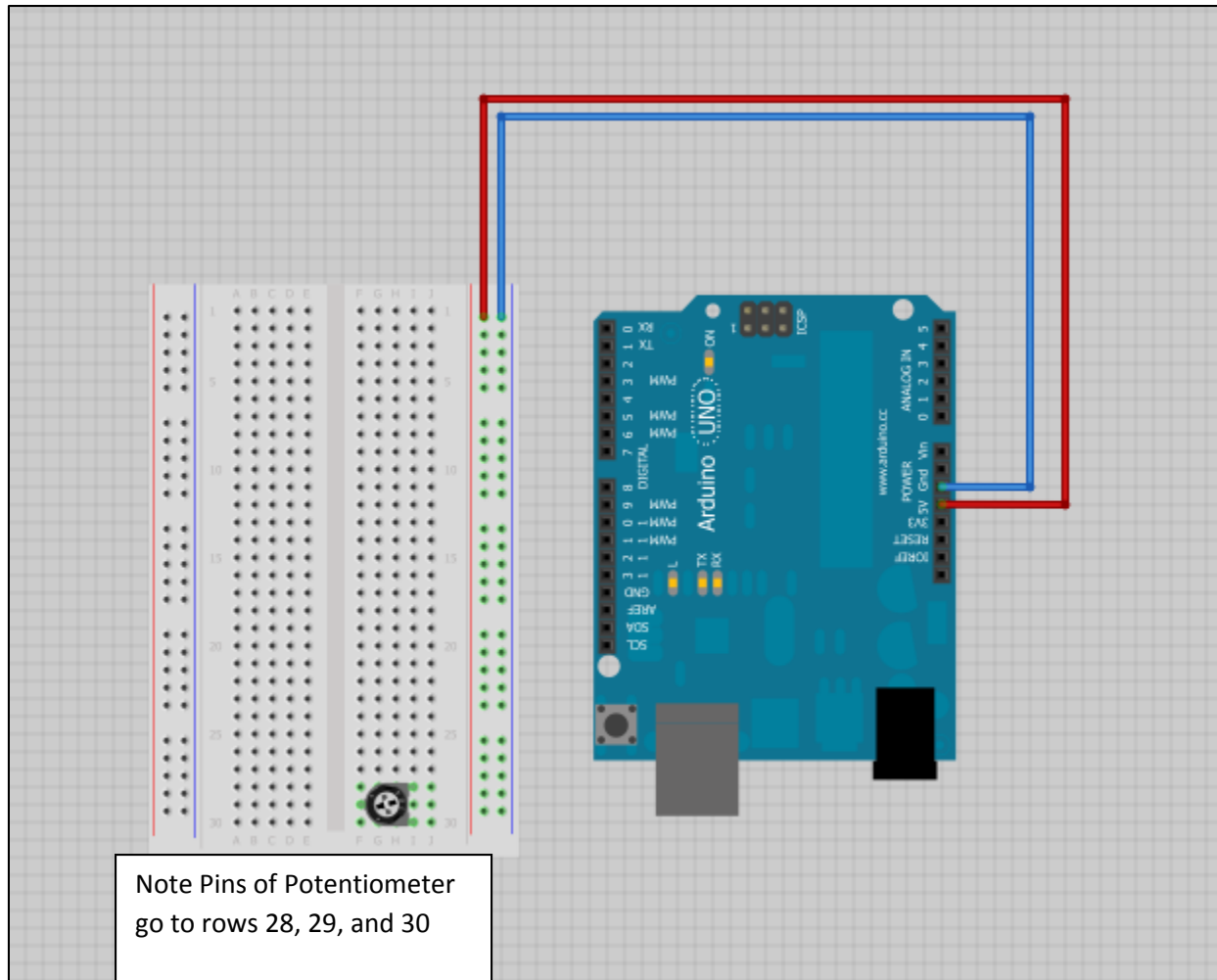
or



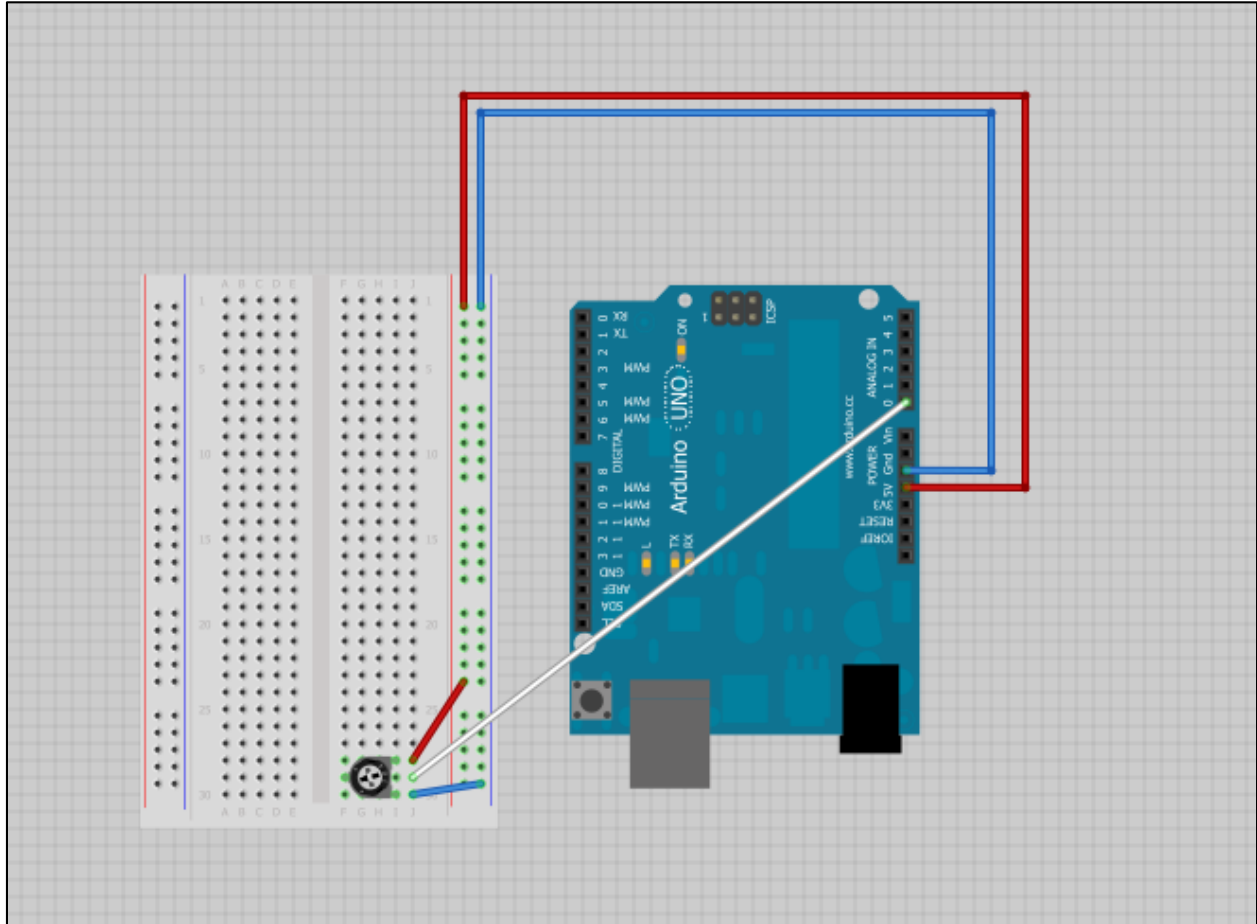
2. First we will run the current and ground wires to the breadboard.
  - a. Connect the red wire from Arduino 5V to the red rail for current.
  - b. Connect the blue wire from Arduino Ground to the blue rail for ground.



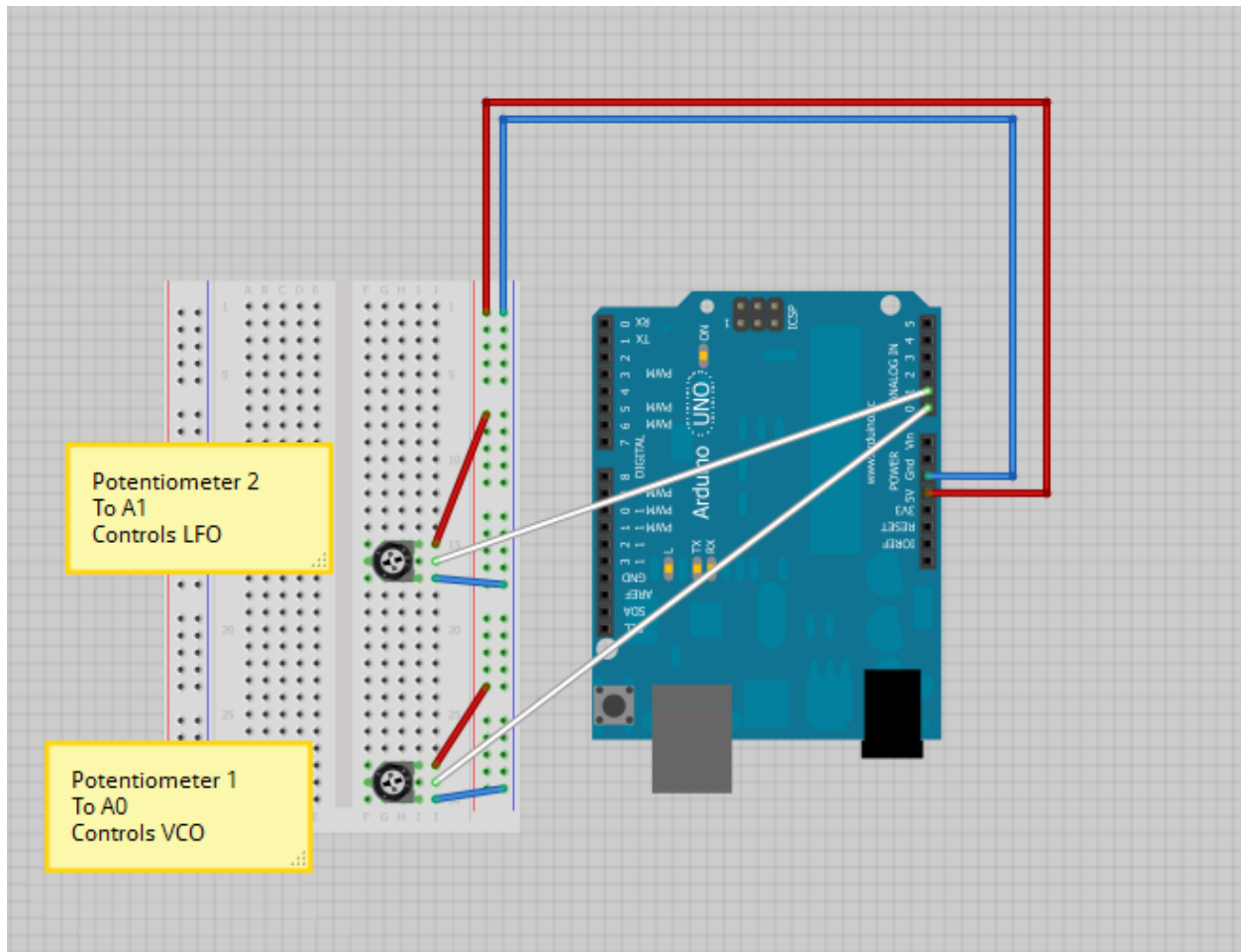
3. Now plug in Potentiometer 1 into breadboard ports 28, 29, and 30 on the F-J side



4. We will now wire the Potentiometer 1.
- a. Run a red wire from breadboard J28 to the red rail for **current**.
  - b. Run a white wire from breadboard J29 to Arduino Analog 0 for **signal**.
  - c. Run a blue wire from breadboard J30 to the blue rail on the Breadboard for **ground**.



5. We will repeat this process with a second potentiometer. Plug a second potentiometer into breadboard rows 15, 16, and 17 in the F to J area.
- a. Run a red wire from breadboard J15 to the red rail for **current**.
  - b. Run a white wire from breadboard J16 to Arduino Analog 0 for **signal**.
  - c. Run a blue wire from breadboard J17 to the blue rail on the Breadboard for **ground**.



## Section 2: Downloading Arduino Software and Setting Up Arduino Board

1. Download the installer software by clicking on:

<http://nebomusic.net/arduinolessons/arduino-1.0.6-windows.exe>

Or:

<http://nebomusic.net/arduinolessons/arduino-1.0.6-windows.zip>

2. Click on the software and run the installation. Select 'Yes' and 'OK' when prompted.

3. With the Arduino board wired we now need to load some code onto the Arduino so it will communicate with Processing. Start the Arduino Sketch program on your computer.



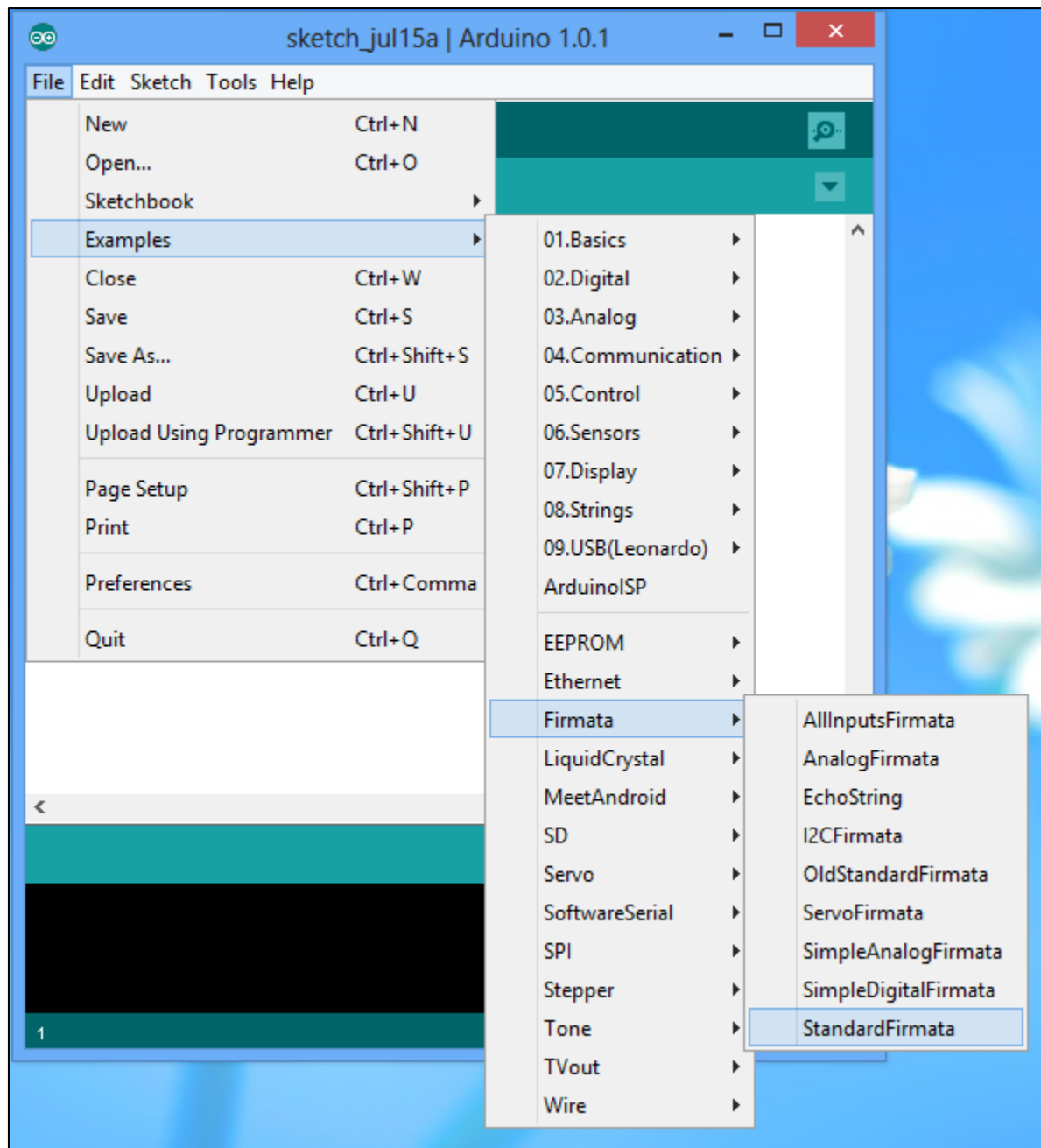
4. Plug in your Arduino Board to your computer with the USB Cable. If you have an Apple OSX computer, select 'Cancel' when prompted to setup a USB modem.

5. Your computer should find the port automatically.

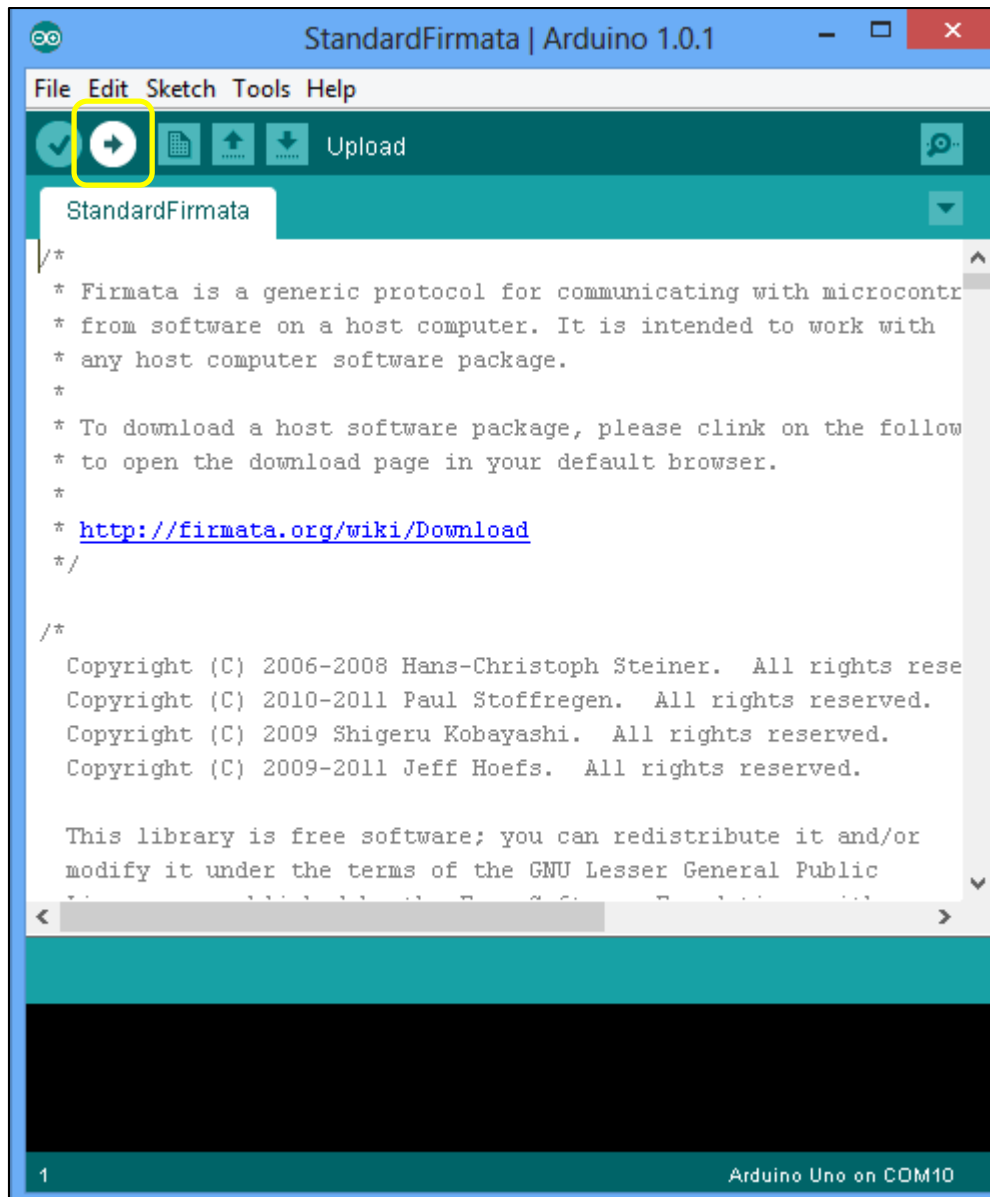
a. In OSX computers go to "Tools-Serial Port" and select `"/dev/tty.usbmodemxxxx"` (The four xxxx will be some type of number)

b. In Windows Computers go to "Tools-Serial Port" and select "COM X" where X will be the com port for the Arduino.

6. With the Arduino communicating with Sketch – Select 'File -> Examples -> Firmata -> StandardFirmata' to open the firmware code.



7. Click the 'Upload' icon to upload this code to your Arduino board.



8. You are finished setting up your Arduino. Move onto the next section to open the Code for the synthesizer simulator.



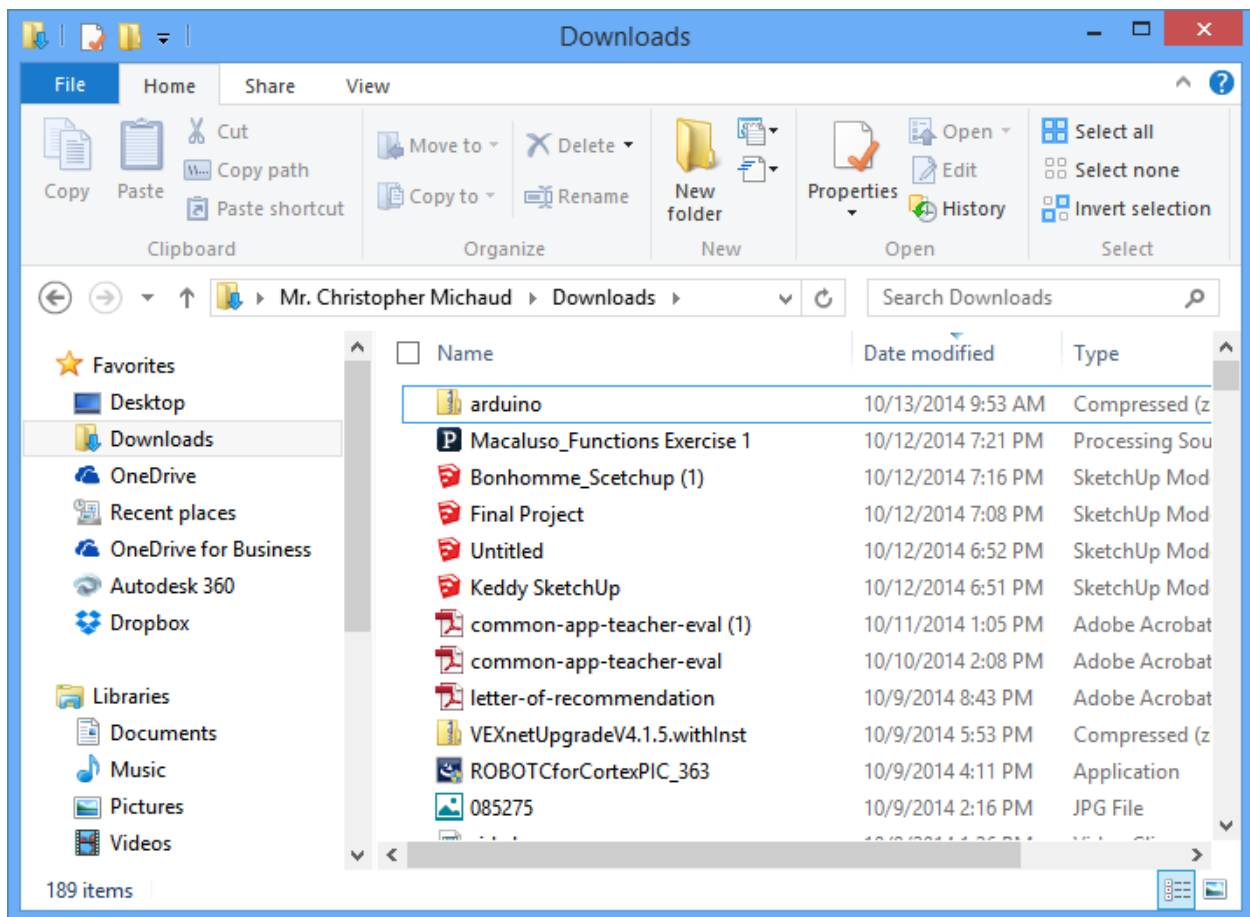
### Section 3: Copying arduino library files to the Documents/Processing/libraries directory

Processing needs a library of files to communicate with the Arduino board. In this section we will download and copy the needed files to the location Documents/Processing/libraries

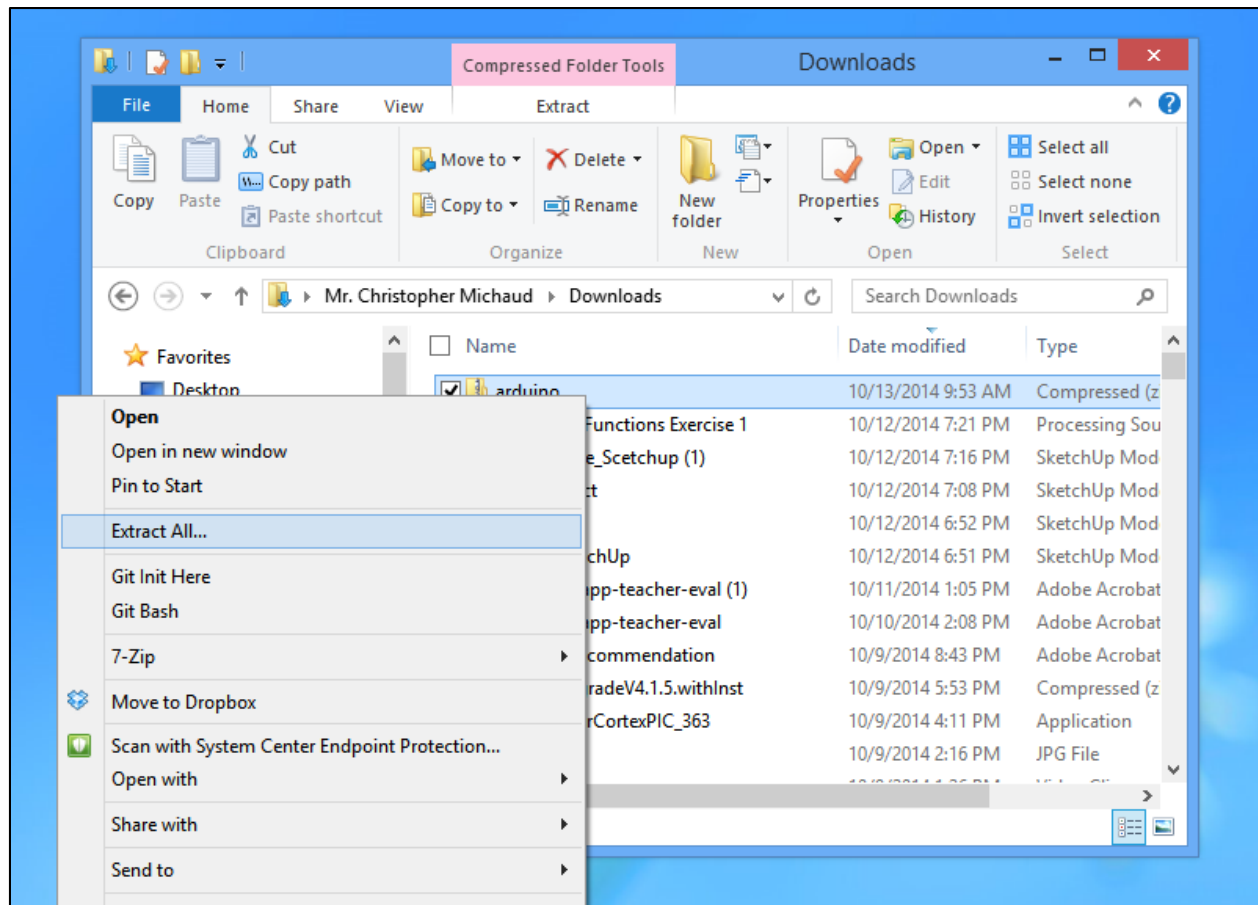
1. Download the arduino.zip file from this link:

<http://nebomusic.net/processinglessons/arduino.zip>

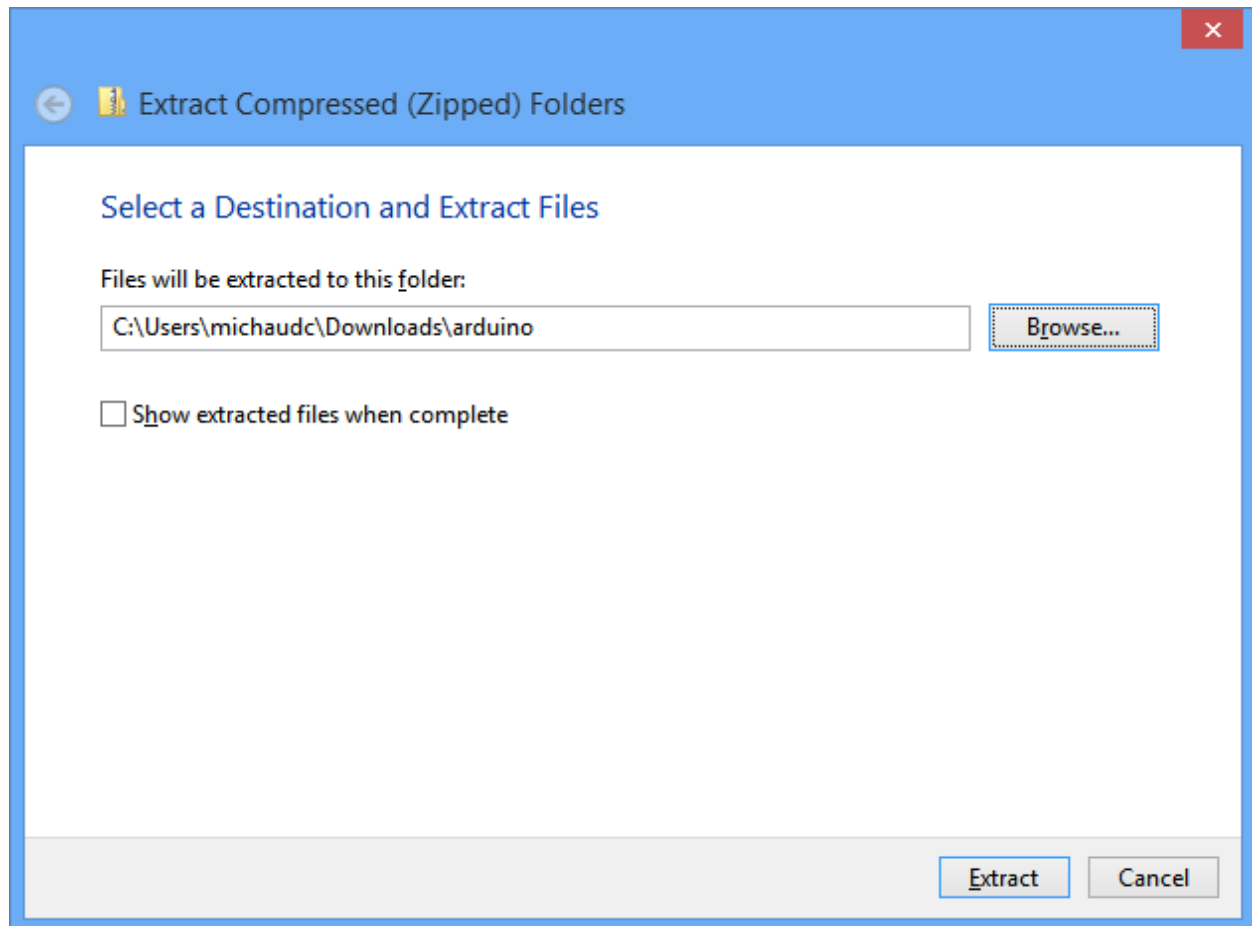
2. Go to your Downloads directory and locate the arduino.zip file.



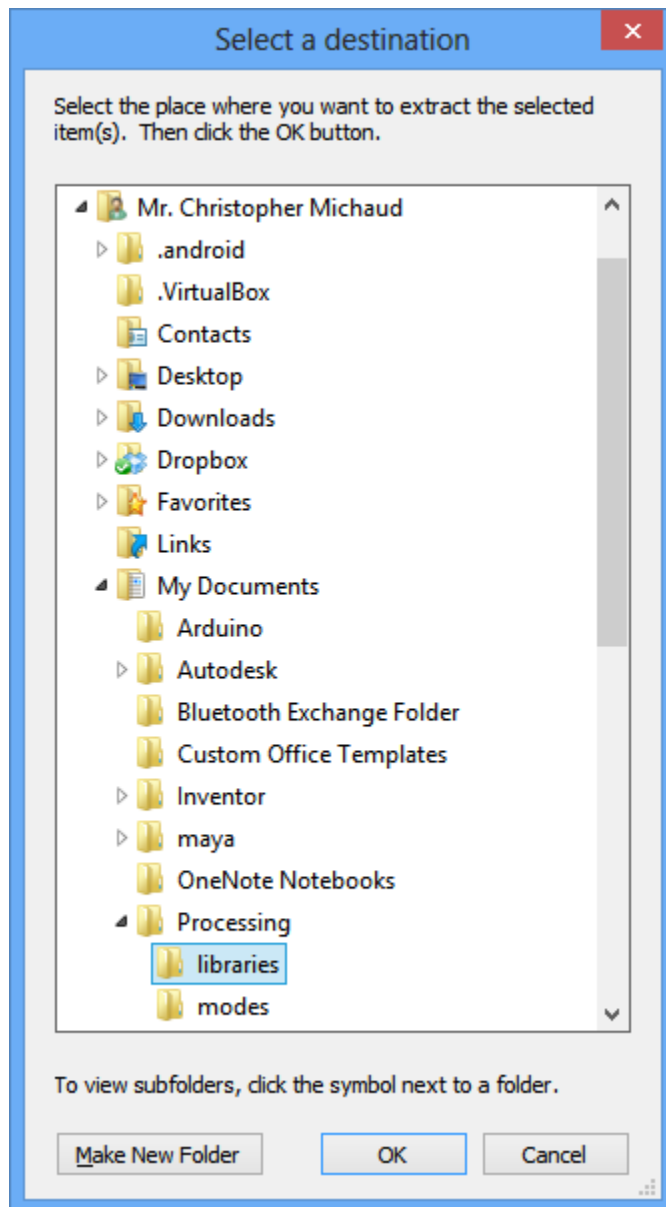
3. Right click on the arduino.zip and select 'Extract All'



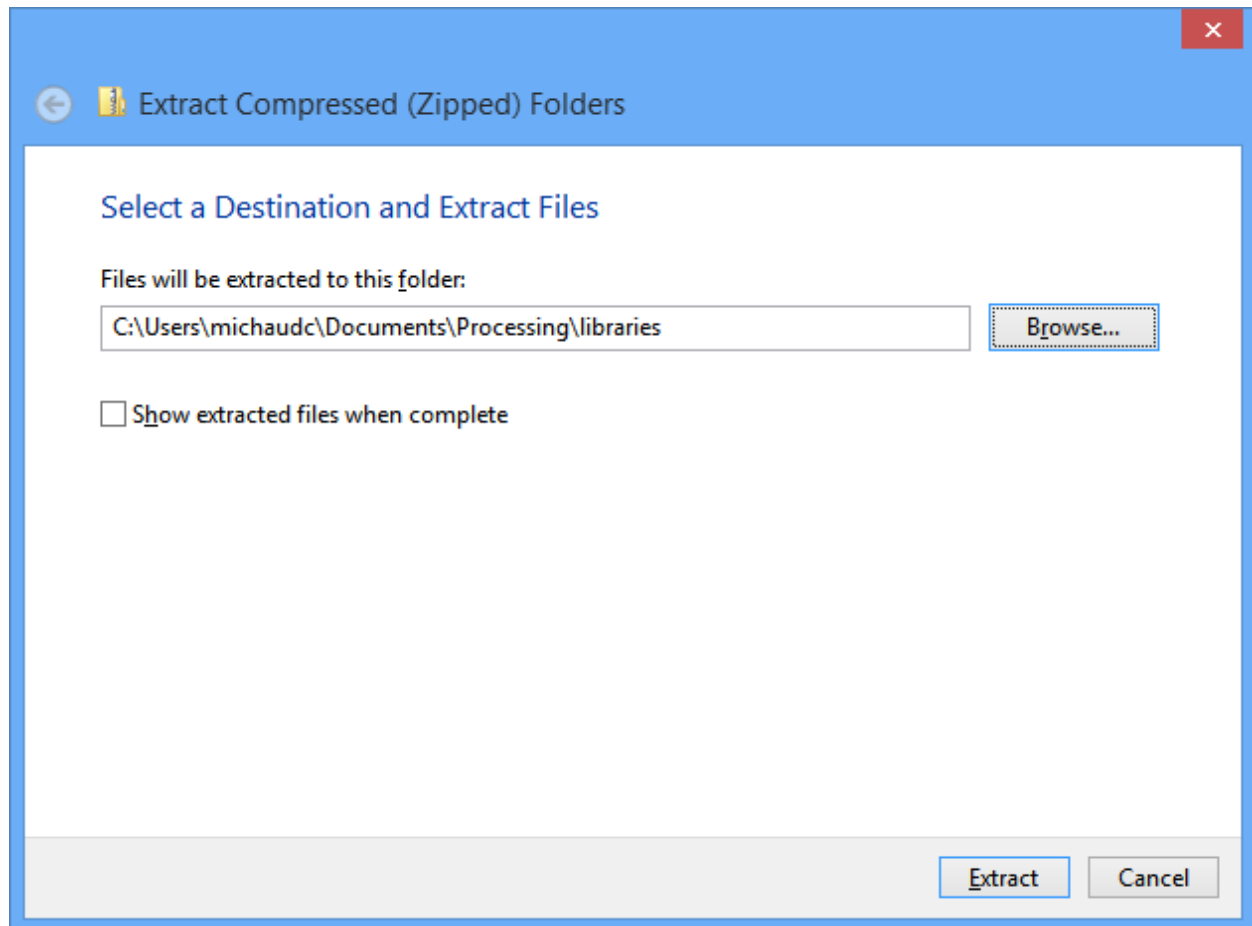
4. Click the 'Browse' button



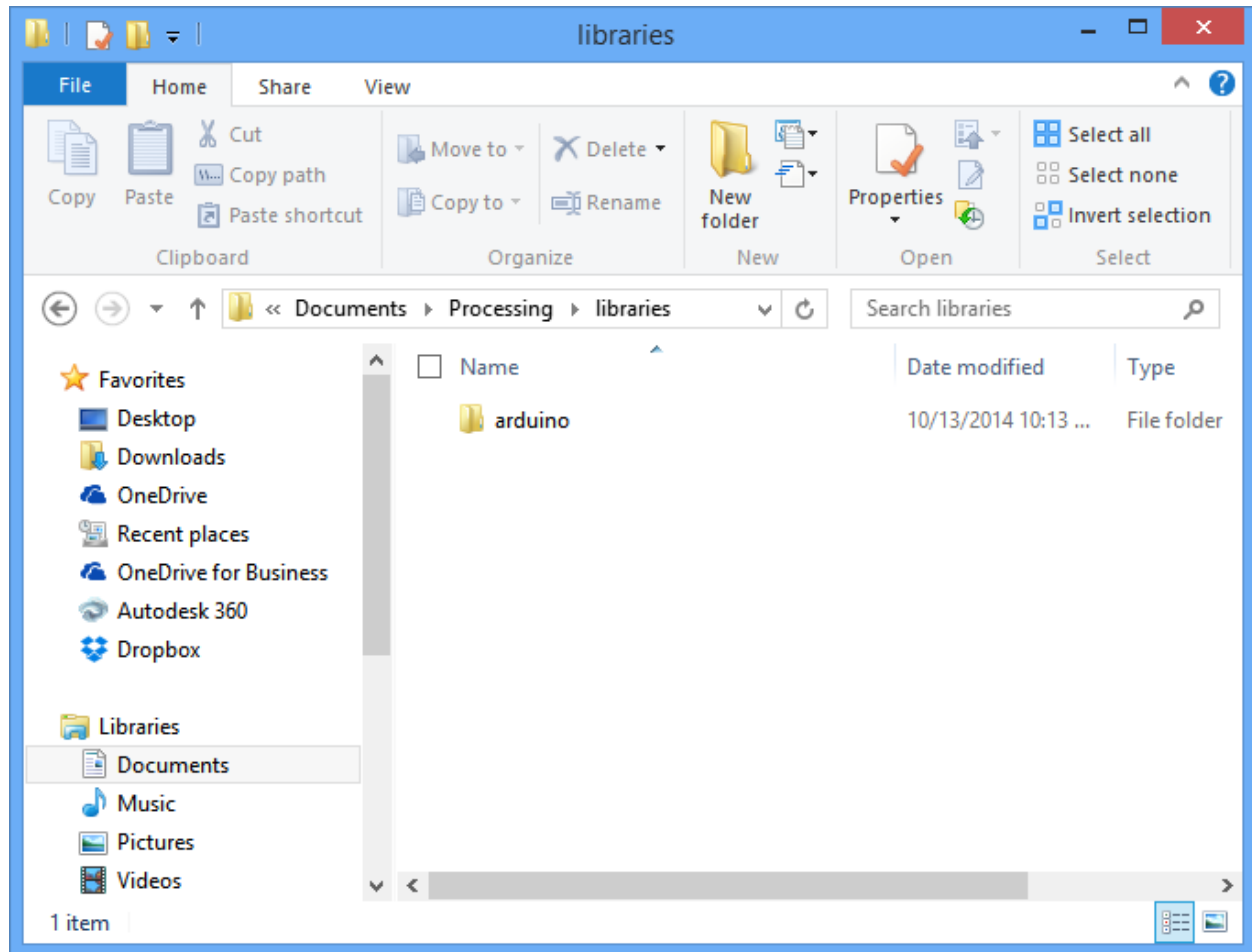
5. Use the 'Select a destination' window and navigate to '<Your Name>/My Documents/Processing/libraries' and then click 'OK'



6. The Extract window should look like this: Click 'Extract'



7. The files will extract. You can check them by going to 'My Documents/Processing/libraries' and you should see the arduino folder.



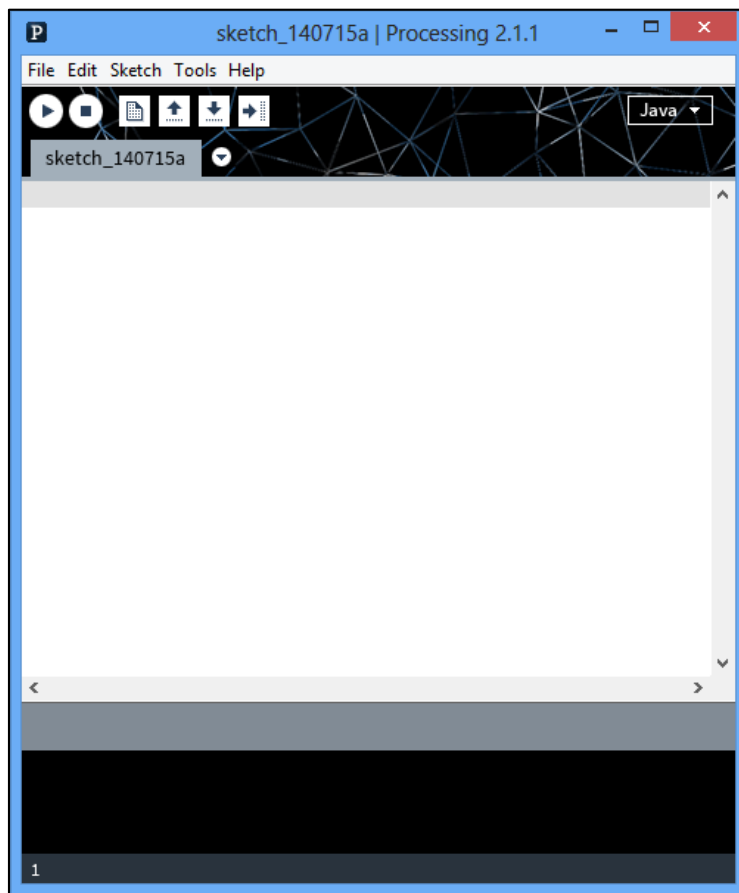
## Section 4: Starting Processing and Loading the Code.

1. Processing is a Java based development environment created at MIT for using code to control graphics, sound, and other media. Start Processing on your computer by clicking the Processing icon.



(You can also click the search tool and type Processing, then click the Processing icon to start)

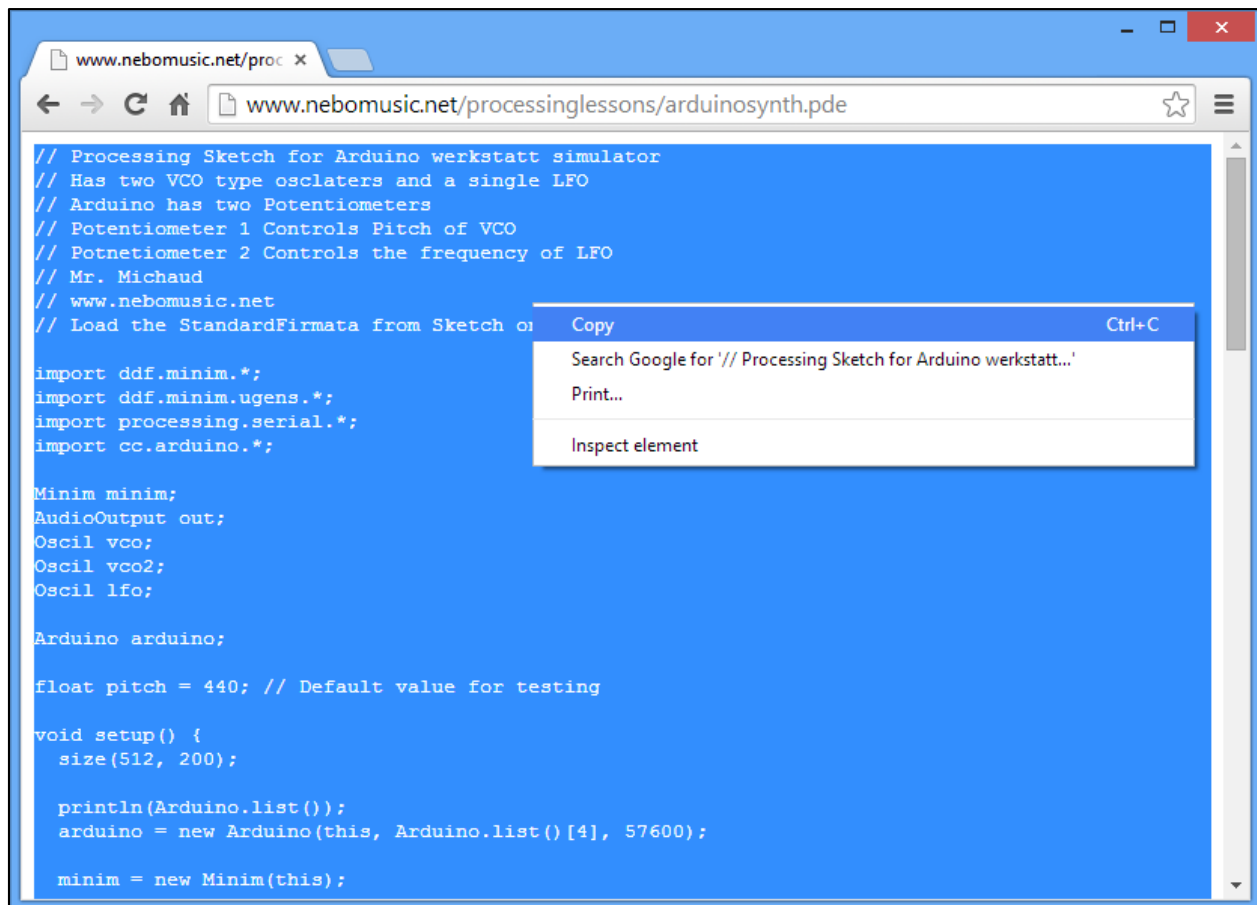
When processing starts it should look like this:



2. We are going to use pre-written code for this project. Open Chrome and go to this URL:

<http://www.nebomusic.net/processinglessons/arduinomusic.pde>

3. Your browser should show a text file with the program. Select all the text and then copy the text.



The screenshot shows a web browser window with the address bar displaying `www.nebomusic.net/processinglessons/arduinomynth.pde`. The main content area has a blue background and contains the following Processing code:

```
// Processing Sketch for Arduino werkstatt simulator
// Has two VCO type osclaters and a single LFO
// Arduino has two Potentiometers
// Potentiometer 1 Controls Pitch of VCO
// Potentiometer 2 Controls the frequency of LFO
// Mr. Michaud
// www.nebomusic.net
// Load the StandardFirmata from Sketch on

import ddf.minim.*;
import ddf.minim.ugens.*;
import processing.serial.*;
import cc.arduino.*;

Minim minim;
AudioOutput out;
Oscil vco;
Oscil vco2;
Oscil lfo;

Arduino arduino;

float pitch = 440; // Default value for testing

void setup() {
  size(512, 200);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[4], 57600);

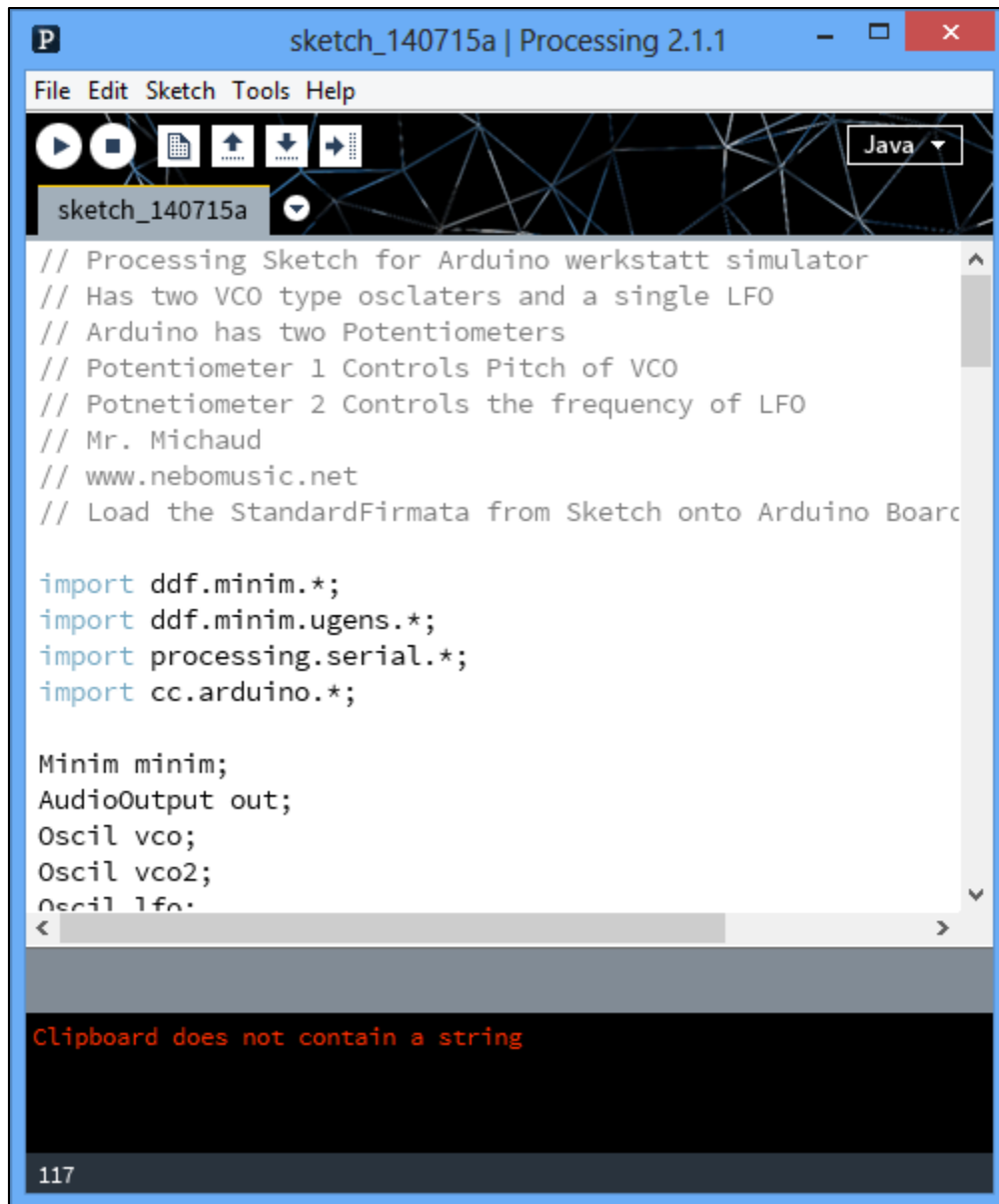
  minim = new Minim(this);
```

A right-click context menu is open over the code, showing the following options:

- Copy (Ctrl+C)
- Search Google for '// Processing Sketch for Arduino werkstatt...'
- Print...
- Inspect element

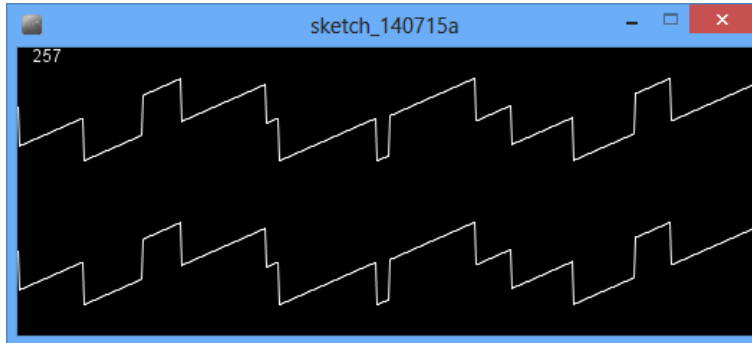


4. Go to the Processing window and paste the code from the website.

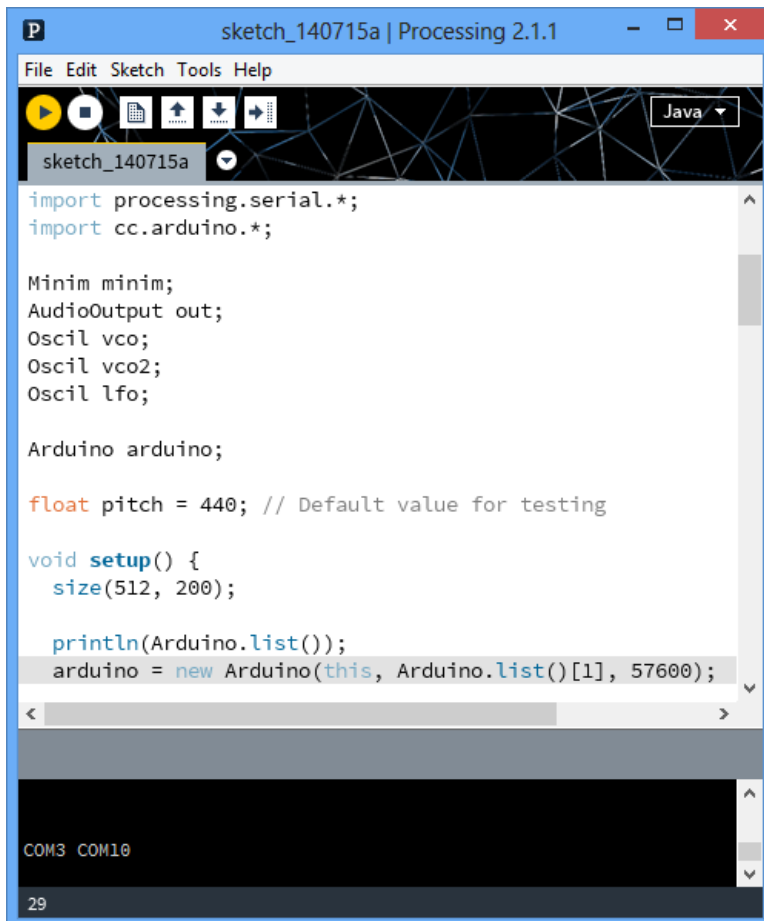


5. Save the file by going "File-Save" and name the file 'arduinomynth'.

6. Make sure the Arduino is plugged into the computer. Click the 'Play' icon to run the program. Press the keys 'a s d f g h j k' to play different pitches. You can adjust the pitch with the Potentiometer 1 and the LFO with Potentiometer 2.



Why two sound waves? These represent the right and left stereo channels of the output to computer sound.



```
import processing.serial.*;
import cc.arduino.*;

Minim minim;
AudioOutput out;
Oscil vco;
Oscil vco2;
Oscil lfo;

Arduino arduino;

float pitch = 440; // Default value for testing

void setup() {
  size(512, 200);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[1], 57600);
}
```

COM3 COM10

29

## 7. Some trouble-shooting tips:

If the program does not run – usually it is because it did not find the right USB device. On line 29 of the Processing program, there is an array called 'Arduino.list' and then a number as an index. In the message window (the black window below the code) there will be a list of USB devices. This list starts counting from zero. Try changing the number in the index to find the match to your USB Arduino connection.



The screenshot shows the Processing IDE interface. The code editor at the top contains the following code:

```
println(Arduino.list());  
arduino = new Arduino(this, Arduino.list()[1], 57600);  
  
minim = new Minim(this);  
  
out = minim.getLineOut();
```

A blue arrow points from the text above to the `Arduino.list()[1]` in the code. Below the code editor is the message window, which is a black area. It displays the output of the code: `0 1` and `COM3 COM10`. The `COM3` and `COM10` text is highlighted in yellow. At the bottom of the message window, the line number `29` is visible.

For an Apple OSX computer – this number is usually 3 or 4.

8. If your synth plays – congratulations! Move on to Section 3 for some tips on how to change the program and create new sounds.

## Section 5: Modifying the arduinosynth program.

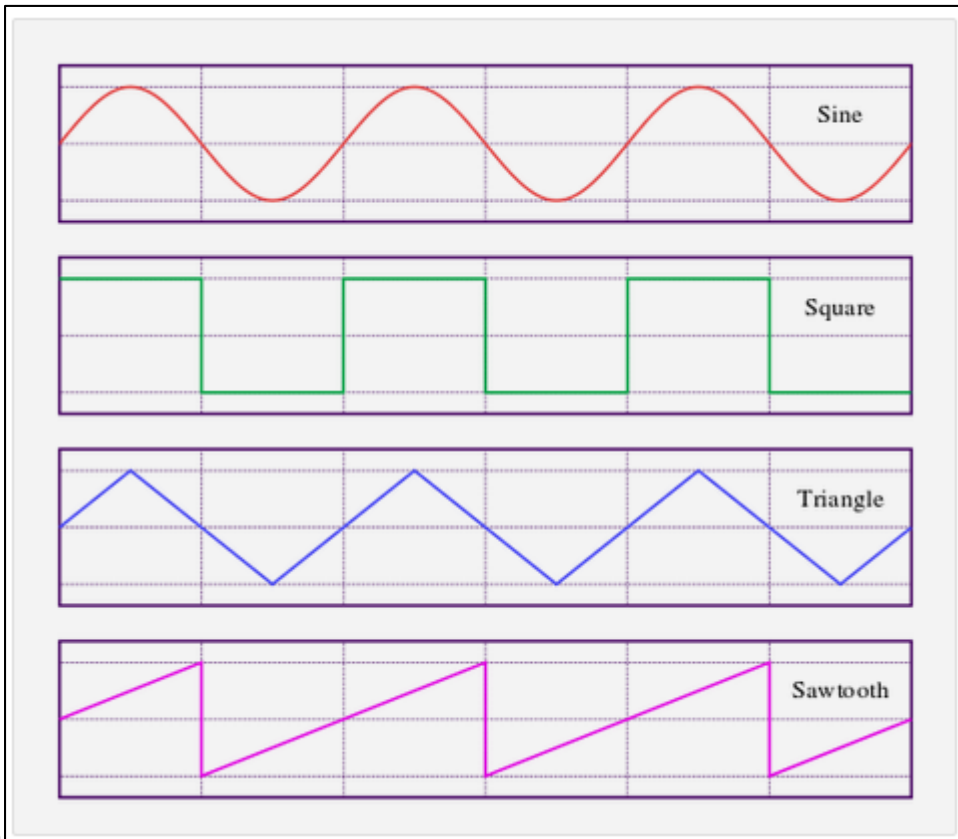
We will make some code modifications to change waveforms and subtract one of the oscillators from the code. The Processing code in this program makes use of the minim library. The documentation can be found at: <http://code.compartmental.net/minim/>. The Javadocs can be found at: <http://code.compartmental.net/minim/javadoc/>

### Part A: Changing Waveforms

1. Find lines 36 and 37 in the code. These two lines initialize vco and vco1 which are two oscillators that provide frequencies for the program.

```
// Create wave oscillators  
vco = new Oscil(pitch, 0.7f, Waves.SQUARE);  
vco2 = new Oscil(pitch, 0.3f, Waves.SAW);
```

2. The constants SQUARE and SAW describe the type of wave. In music synthesis we have different waveforms:

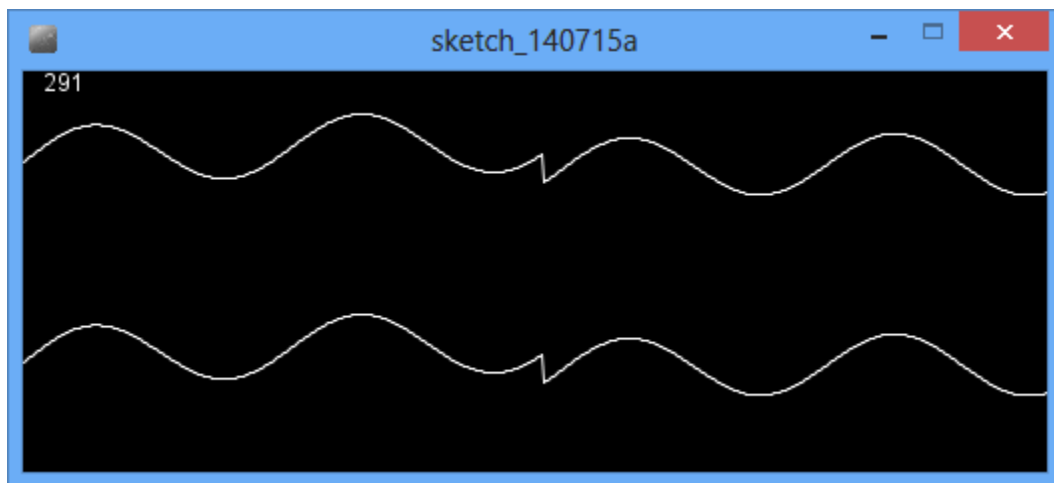


3. In Minim we have the following Wavforms:

SINE  
SAW  
PHASOR  
SQUARE  
TRIANGLE

4. Experiment with lines 36 and 37 changing the Waves.SQUARE value to another waveform.  
For Example:

```
// Create wave oscillators  
vco = new Oscil(pitch, 0.7f, Waves.PHASOR);  
vco2 = new Oscil(pitch, 0.3f, Waves.SINE);
```



## Part B: Removing the Overtone (vco2)

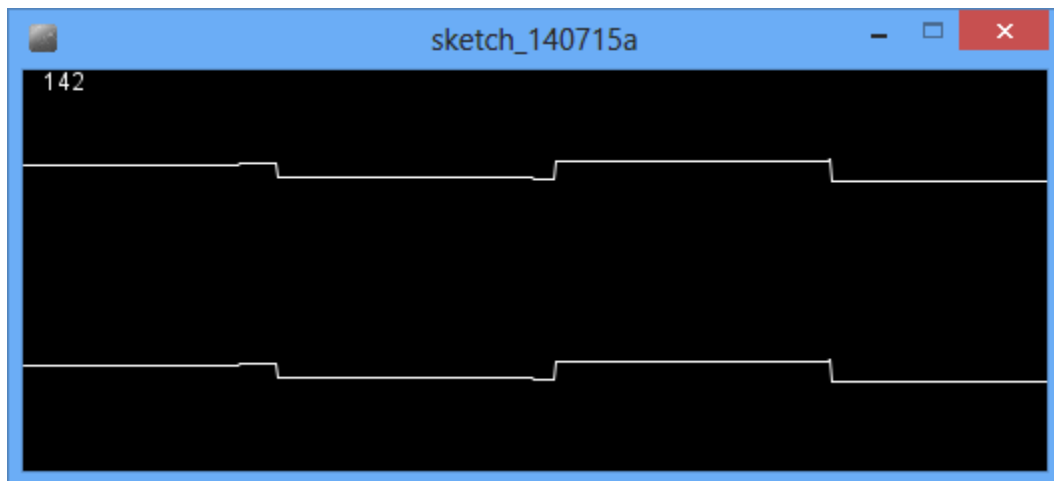
1. There are two oscillators for pitch (vco and vco2). We are going to comment out vco2 so we can hear only the vco. Find line 46 in the code:

```
vco.patch(out);  
vco2.patch(out);
```

2. Place two slash marks (//) before the code `vco2.patch(out)`. This comments out the line like the # does in python. Commenting out this line will remove vco2 from the output. (This is like muting a channel).

```
vco.patch(out);  
// vco2.patch(out);
```

3. Run the code and note the difference in the sound. You can also modify line 36 with the waveform to hear how the different shapes of the waves change the timbre.



### Part C: Adding another Oscillator(vco3)

1. We are now going to add another oscillator unit to the mix. First, uncomment line 46 so vco2 will be patched to the output sound.

```
vco.patch(out);  
vco2.patch(out);
```

2. Find line 18 in the code and press enter to put a blank line in 19.

```
Minim minim;  
AudioOutput out;  
Oscil vco;  
Oscil vco2;  
  
Oscil lfo;
```

3. The section of code from lines 15 to 20 define objects from the Minim, Arduino, and Oscil classes. We want another Oscil object (Oscil stands for 'Oscillate' or vibrate). Type

Oscil vco3;

in line 19:

```
Minim minim;  
AudioOutput out;  
Oscil vco;  
Oscil vco2;  
Oscil vco3; // New Oscil object  
Oscil lfo;
```

4. Find line 38 and press enter to make a new line:

```
// Create wave oscillators  
vco = new Oscil(pitch, 0.7f, Waves.SQUARE);  
vco2 = new Oscil(pitch, 0.3f, Waves.SAW);
```

5. Type:

```
vco3 = new Oscil(pitch, 0.5f, Waves.SINE);
```

in line 38.

```
// Create wave oscillators  
vco = new Oscil(pitch, 0.7f, Waves.SQUARE);  
vco2 = new Oscil(pitch, 0.3f, Waves.SAW);  
vco3 = new Oscil(pitch, 0.5f, Waves.SINE);
```

6. We need to patch the vco3 to the output. Type:

```
vco3.patch(out);
```

In a new line 49:

```
vco.patch(out);  
vco2.patch(out);  
vco3.patch(out);
```

7. Finally, we need to set the pitch of vco3 by setting the frequency. We are going to relate the frequency to a variable pitch in the program and then multiply to get an overtone. Find line 60.

```
// Set Pitch  
vco.setFrequency(pitch);  
vco2.setFrequency((pitch*1.2599)*2); // Overtone at the Major 10th
```

8. Type the following into line 60: (Note the use of comment)

```
vco3.setFrequency((pitch*1.5)*); // Overtone at Perfect 12th
```

```
// Set Pitch  
vco.setFrequency(pitch);  
vco2.setFrequency((pitch*1.2599)*2); // Overtone at the Major 10th  
vco3.setFrequency((pitch*1.5)*2); // Overtone at Perfect 12th
```



9. Run the code and note the difference in sound. Experiment by changing the Waveforms or Frequencies of the 3 oscillator objects. Instrument timbres are combinations of frequencies, timbres and overtones. With a digital or analog synthesizer, we can create an infinite variety of sounds!

The Moog Werkstatt is an Analog synthesizer featuring the same units as we simulated in processing (VCO, LFO) plus some other units. The Moog synthesizers also work in a similar manner.

